REMARKS/ARGUMENTS

In the Office Action, the Examiner noted that claims 1-23 are pending in the application. The Examiner additionally stated that claims 1-23 are rejected. By this amendment, claims 1, 8, and 18 are amended. Hence, claims 1-23 are pending in the application.

Applicant hereby requests further examination and reconsideration of the application, in view of the foregoing amendments.

In the Specification

Applicant has amended the specification to secure a substantial correspondence between the claims amended herein and the remainder of the specification. No new matter is presented.

In the Claims

Rejections Under 35 U.S.C. §112

The Examiner rejected claims 1-7 under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. It was noted that the scope of meaning of claim 1 is not dear because it ends in a semicolon ";" and, therefore also the scope of meaning of the claims that depend on claim 1 is also unclear.

By this communication, claim 1 is amended to end in a period so that its meaning, and the meaning of dependent claims 2-7, are clear.

Accordingly, Applicant respectfully requests that the examiner withdraw the rejections of claims 1-7.

Rejections Under 35 U.S.C. §103(a)

The Examiner rejected claims 1-23 are rejected under 35 U.S.C. 103(a) as being unpatentable over Knebel, U.S. Patent No. 6,618,801 (hereinafter, Knebel), in view of Blomgren, U.S. Patent No. 5,695,009 (hereinafter, Blomgren).. Applicant respectfully traverses the Examiner's rejections.

With respect to claims 1, 7, 8, and 18, the Examiner noted that Knebel taught the invention substantially as claimed including a data processing ("DP") system comprising (As per claims 1, 7 ,8,18):

a) Apparatus in a microprocessor for executing native instructions (e.g. see figs. 1-3 and col. 1, lines 34-61);

b) Instruction translation logic (fetch engine 20) (e.g., see figs. 1, 2, 3, and col. 1, lines 34-61), configured to retrieve macro instructions provided, and configured to translate each of the macro instruction into associated native instructions for execution, wherein if a first form of the instruction is retrieved, the instruction translation logic (20) directs the microprocessor to enable native bypass mode and indicates such by asserting a first bit (predecode bits) (e.g., see figs. 1, 2, 3) and col. ?, 46-64);

c) Bypass logic (60,70), coupled to the instruction translation logic (201), configured to access a control bit (pre-decode bits) to determine if the native bypass mode is been enabled and to detect wrapper macro instructions and, upon detection of the wrapper macro instructions, to provide the native instructions for execution by the microprocessor, thereby bypassing the instruction translation logic,(e.g., see fig. 2 and col. 1, lines 36-61 and col. 2, lines 29-56) [The Examiner noted that the fetch engine decodes the instruction that are macro instructions into microinstructions and directs emulation engine to bundle instructions and fetch engine separately sends instructions that are already microinstructions when fetched to multiplexer (70) [the logic within fetch unit that determined whether the instruction is a macroinstruction that would need decoding or already a microinstruction when fetch transmits the instruction that were already microinstructions and the one decoded via separate conductors one to the emulation engine and the other directly to the multiplexer provides for the claimed bypass operation.

The Examiner further stated that Knebel did not expressly detail that the translation logic was disabled, but that one of ordinary skill would have been motivated to send each incoming instruction code via two parallel pipelines one for macroinstructions and one for instructions that were already microinstructions to take advantage of the possibility of

the instruction already being a microinstruction that only needed to be forwarded increasing speed of production microinstructions. In order to do this when the instruction was a microinstruction it would have been input to the macroinstruction decoder within the fetch unit. At that time one of ordinary skill would have been motivated to disable the decoder for translating the instruction and producing output because the decoding would produce an incorrect result and/or be redundant. All instructions whether instructions or microinstructions were in the same pipeline then the instructions that were already microinstruction would have to unnecessarily wait. Even if this type of system were employed when a microinstruction was input to the decoder an incorrect result would be produced and unnecessary delay would have been encountered.

The Examiner noted in addition that Knebel did not expressly detail (1, 8, 11, 14, 19, 22) asserting a first bit within a control register, but that Knebel did specify predecode bits (e.g., see col. 1, lines 36-61). It was further pointed out that Blomgren taught when an instruction is not supported in the CISC instruction set signaling an exception and asserting a bit in a flags register to change to another mode for changing from instruction set types one of which was high level CISC and the other comprised microinstructions RISC (e.g., see col. 8,line 53-col. 9, line 29 and col. 10, lines 18-39). In this situation the emulation of CISC instructions using RISC instructions would have provided the microinstructions to the fetch unit in Knebel that would have by passed the decoder in the fetch unit (that translates macroinstructions). The Examiner opined that when the system finished executing the not instruction one of ordinary skill would have been motivated to reset the control bit for returning to CISC processing to process the remaining instructions within the program. Also since Blomgren taught setting flags bit for changing to a mode one of ordinary skill would have been motivated to check the register (FLAGS register) to determine the mode the system is in.

Regarding claims 2 and 9, the Examiner noted that as to the embedding of the native instructions within wrapper, sending of RISC instructions to the for execution and taught by Blomgren meets this limitation to the extent claimed, stating that, in addition Blomgren and Knebel taught a processor that can execute RISC and CISC programs. Knebel taught combining CISC and RISC registers. (e.g.,. see col. 15, line 13-col. 57),

and that , therefore, in order to take advantage of availability of CISC instruction registers one of ordinary skill would have been motivated to format the RISC instructions for storage in the CISC registers. The Examiner pointed out that this would have provided an embedded instruction (e.g., see col. 15, lines 27-col. 16, line col. 17, lines 67).

Applicant respectfully disagrees with the Examiner's characterization of the teachings of Knebel and Blomgren, and furthermore with those teachings in terms of the teachings of the instant application.

Knebel teaches a method for implementing two architectures on a single chip. The method uses a fetch engine to retrieve instructions. If the instructions are macroinstructions, then it decodes the macroinstructions into microinstructions, and then bundles those microinstructions using a bundler, within an emulation engine. The bundles are issued in parallel and dispatched to the execution engine and contain pre-decode bits so that the execution engine treats them as microinstructions. Before being transferred to the execution engine, the instructions may be held in a buffer. The method also selects between bundled microinstructions from the emulation engine and native microinstructions coming directly from the fetch engine, by using a multiplexor or other means. Both native microinstructions and bundled microinstructions may be held in the buffer. (Abstract). When executing in native mode, the fetch engine 20 delivers 32 bytes of instruction stream to the execution engine 40. Within each 16 bytes, or "bundle," the RISC ISA so defines there to be three 41-bit instructions and five bits of template information. In addition, the fetch engine 20 sends other control information, called pre-decode bits, that it decodes from the 16 bytes of the instruction stream. The pre-decode bits are used by the execution engine 40 to help it efficiently distribute the six instructions to the proper execution units. When executing in emulation mode, it is necessary for the execution engine 40 to receive data in exactly the same format as it does in native mode. This allows the vast majority of the execution engine 40 to be designed only for native mode execution, while allowing it also to be used when in emulation mode. Thus, the emulation engine 60 must also deliver 32 bytes of instruction data along with the pre-decode bits calculated from those 32 bytes. (col. 2, lines 47-64)

Clearly, Knebel teaches a processor that is capable of directly executing microinstructions provided by a fetch engine and microinstructions that are delivered via an emulation engine, where those delivered via the emulation engine have been translated from CISC macroinstructions. Knebel also refers to a native mode and to an emulation mode. But nowhere does he teach how the microprocessor changes from one mode to the other. He is completely silent with regard to mode changes. Is Knebel's processor capable of automatically switching between these two modes, or must the processor be put in a native mode and then exclusively supplied native instructions to execute, or be put in an emulation mode and then exclusively supplied with macro instructions? Knebel does not suggest that his processor can distinguish between RISC and CISC instructions either. All that Knebel teaches is that, when in native mode, microinstructions are provided by the fetch engine to the execution engine, and when in emulation mode, the emulation engine provides the microinstructions. The Examiner's suggestion that the logic within fetch unit determines whether an instruction is a macroinstruction that would need decoding or already a microinstruction when fetched is only true insofar as the fetch engine is directed to do so via knowing which mode it is operating in.

Knebel does not teach directing a microprocessor to enable a native bypass mode upon retrieval of a first form of a first macro instruction. Knebel does not discuss, or even suggest, how such a mode is enabled or disabled. Knebel does not teach indicating a native bypass mode by asserting a first bit in a control register. The Examiner's assertion that Knebel's predecode bits are equivalent to asserting a first bit in a control register to indicate native bypass mode is unsupported by the reference. Again, Knebel states that predecode bits are part of what is to be supplied to the execution engine, and these must be supplied as part of the instruction stream by the fetch engine, *when in native mode.* Applicant's invention, in contrast, indicates native bypass mode by asserting the first bit in a control register.

In addition, Knebel's bypass logic 60, 70 does not access predecode bits to determine if native mode is enabled. Rather, the execution engine of Knebel must be supplied with predecode bits as part of the instruction stream, regardless of which mode it is operating in. Applicant's bypass logic, in contrast, accesses a bit within a control register (having

been previously set by instruction translation logic upon retrieval of a first form of a first macro instruction) to determine if said native bypass mode has been enabled.

Knebel does not teach detection of wrapper macro instructions and, upon detection of said wrapper macro instructions, to disable said instruction translation logic, and to provide the native instructions for execution by the microprocessor, thereby bypassing said instruction translation logic. Knebel furthermore does not teach that wrapper macro instructions are existing macro instructions which are translated by the instruction translation logic according to architectural specifications if the native bypass mode has not been enabled.

Blomgren teaches a technique for merging architecturally defined floating point registers in a CISC instruction set with architecturally defined registers in a RISC instruction set to that the merged registers are shared by CISC and RISC instructions. For example a CISC flags register is merged according to Blomgren with a RISC flags register so that CISC and RISC flags having the same function are merged to the same register bit. (Abstract) The Examiner noted that Blomgren taught when an instruction is not supported in the CISC instruction set signaling an exception and asserting a bit in a flags register to change to another mode for changing from instruction set types, one of which was high level CISC and the other comprised microinstructions RISC (e.g., see col. 8,line 53-col. 9, line 29 and col. 10, lines 18-39). That Blomgren employs a bit in a flags register to indicate that his microprocessor cannot function in one mode and must be changed to another mode does not equate to setting a bit to indicate native bypass mode and then subsequently accessing that bit to perform subsequent functions. The Examiner's assertion that when the system finished executing the not instruction one of ordinary skill would have been motivated to reset the control bit for returning to CISC processing to process the remaining instructions within the program and that since Blomgren taught setting flags bit for changing to a mode one of ordinary skill would have been motivated to check the register (FLAGS register) to determine the mode the system is in. Blomgren does not teach what is asserted by the examiner. What Blomgren teaches in these sections is that since the EFLAGS register is combined, then RISC instructions that emulate CISC instructions, where the CISC instructions update a bit in EFLAGS, must in

fact update the same bit in EFLAGS, due to the merging of the RISC and CISC EFLAGS registers into a combined register. Similarly, when the merged register is read to determine architectural operating state (e.g., virtual 8086 mode, interrupt enables, etc.), it must be interpreted in the same manner by both RISC and CISC instructions, due to the merge. In essence, Blomgren adds nothing to the issues at hand.

Accordingly, in view of the above points and arguments, it is respectfully requested that the rejections of claims 1, 8, and 18 be withdrawn.

With regard to claims 2-7, 9-17, and 19-23, these claims depend from claims 1, 8, and 18, respectively and add further limitations that are neither anticipated nor made obvious by Knebel, Blomgren, or Knebel and Blomgren in combination. Accordingly, Applicant respectfully requests that the Examiner withdraw the rejections of claims 2-7, 9-19, and 19-23.

CONCLUSIONS

In view of the arguments advance above, Applicant respectfully submits that claims 1-23 are in condition for allowance. Reconsideration of the rejections is requested, and allowance of the claims is solicited.

Applicant earnestly requests that the Examiner contact the undersigned practitioner by telephone if the Examiner has any questions or suggestions concerning this amendment, the application, or allowance of any claims thereof.

I hereby certify under 37 CFR 1.8 that this correspondence is being facsimile transmitted to the United States Patent and Trademark Office on the date of signature shown below.

Respectfully submitted,
HUFFMAN PATENT GROUP, LLC

/Richard K. Huffman/

By: _____

RICHARD K. HUFFMAN, P.E.
Registration No. 41,082
Tel: (719) 575-9998

06/23/2006

Date: _____